

Unlink <modname> Usage : Unlinks module(s) from memory @WCREATE
 Syntax: Wcreate [opt] or /wX [-s=type] xpos ypos xsiz ysiz fcol bcol [bord]
 Usage : Initialize and create windows Opts : -? = display help -z = read

command
 lines from
 stdin -s=type
 = set screen
 type for a
 window on a

AUSTRALIAN OS9 NEWSLETTER

new screen
 @ X M O D E
 Syntax :
 X M o d e
 <devname>
 [params]

Usage : Displays or changes the parameters of an SCF type device
 @COCOPR Syntax: cocopr [<opts>] {<path> [<opts>]} Function: display file
 in specified format gets defaults from /dd/sys/env.file Options : -c set columns
 per page -f use form feed for trailer -h=num set number of lines after
 header -l=num set line length -m=num set left margin -n=num set starting
 line number and incr -o truncate lines longer than lnlen -p=num set number
 of lines per page -t=num number of lines in trailer -u do not use title

EDITOR

Gordon Bentzen

(07) 344-3881

SUB-EDITOR

Bob Devries

(07) 372-7816

TREASURER

Don Berrie

(07) 375-1284

LIBRARIAN

Jean-Pierre Jacquet

(07) 372-4675

Fax Messages

(07) 372-8325

SUPPORT

Brisbane OS9 Users Group

-u=title use specified title -x=num set starting page number -z[=path] read file
 names from stdin or <path> if given @CONTROL Syntax: control [-e] Usage
 : Control Panel to set palettes, mouse and keyboard parameters and monitor

ADDRESSES

Editorial Material:

Gordon Bentzen

8 Odin Street

SUNNYBANK Qld 4109

Library Requests:

Jean-Pierre Jacquet

27 Hampton Street

DURACK Qld 4077

type for
 Multi-Vue.
 Selectable from
 desk utilities
 menu as the
 Control Panel.
 Opts : -e =
 execute the
 environment file
 @ G C L O C K
 Syntax: gclock
 Usage : Alarm
 clock utility for
 Multi-Vue.

CONTENTS

Editorial	Page 2
Floptical Drives...	Page 3
C Tutorial.....	Page 3
MPPM.....	Page 4
Interleave Tester..	Page 6
ShellPlus Scripts..	Page 7

Selectable from desk utilities menu as Clock. @GCALC Syntax: gcalc Usage :
 Graphics calculator utility for Multi-Vue. Selectable from desk utilities menu as

Volume 7

April 1993

Number 3

Calculator. @GCAL Syntax: gcal Usage : Calendar/Memo book utility for
 Multi-Vue. Selectable as Calendar from the desk utilities menu. @GPRINT
 Syntax: gprint Usage : Printer setup utility for Multi-Vue Lets user graphically

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 7 Number 3

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries

TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet

SUPPORT : Brisbane OS9 Level 2 Users Group.

Chicago CoCoFest countdown

Boy doesn't time fly! Here we are in April '93 with Easter almost here and it seems like it was only yesterday or so that we had trouble writing 1993 and often slipped into writing 1992 instead.

The planned CoCoFest in Chicago which was way off in the future not long ago is almost here. May 1st and 2nd 1993 are the dates. My plans to attend as the Australian OS9 Usergroup representative are now finalised with air fares booked and paid for, plus the required passport and visa arrangements completed.

National OS9 Usergroup support.

The following National OS9 Usergroup members have made a financial contribution towards the cost of this trip and we extend our sincere thanks for this support.

ERIC BAILEY - Victoria

GEORGE McLINTOCK - ACT.

Our attendance at this CoCoFest will provide the Usergroup with an opportunity to get the latest information on what is going on in the CoCo community and of course the latest in 6809 (6309) OS9 and OSK. We expect to extend our contacts for public domain software as well as commercial offerings which will be invaluable for the future. This is an opportunity which we just can't afford to miss out on. It is not too late to send your contribution.

PROPOSED INTERNATIONAL COOPERATION

In the editorial last month I gave some details of the meeting planned by Peter Tutelaers to be held on Saturday night May 1st at the same venue as the Chicago CoCoFest. I received a letter this week from Jim DeStafeno of the OS-9 Users Group, USA, with comments on the international "co-operation" proposal. Suffice to say at this point that Jim seems to have quite a different point of view on the way things should be organised. This meeting promises to be very interesting and I would really like to be able to put forward the views of our own usergroup.

To do this of course I need your input and we have only a few weeks left. So please let me have your thoughts on a "International Usergroup" which would

be responsible for collection and distribution of information, PD software etc to all participating usergroups around the world. How would you suggest this be set up, controlled and funded? What benefits to OS9 users could we expect or require?

We in Australia rely on contact with USA where most information and software originates. We also enjoy an exchange arrangement with the EUROS9 usergroup in Europe but a lot of their material is also from the States.

We obtain PD disks from USA and EUROS9 from time to time and have also accessed PD archive files etc from Internet (no longer available to us), CompuServe and Delphi. There are of course costs involved in obtaining this material for our own PD library which is available to all members. As an example, my account for the month of February for using Delphi via Sprintnet is A\$160. Downloading AR files from Delphi at 2400 baud soon runs away with the dollars and really is just too expensive. Disks via the snail mail is far far cheaper, if you can wait that is.

We also take the OS9 echo through the FIDO network which keeps us in touch with the message base used mainly by US and Canadian OS9'ers, but this costs us \$40 per quarter. It seems that the Fido echo is not available, or not taken, by some countries and as such this limits the OS9 user base. We have not yet arranged for file transfer facilities through the Echo as more costs are involved and we would need to find a sysop willing to arrange it. By the way, this OS9 FIDO Echo comes into Sydney (hub) from overseas and is available for other states within Australia. We take it from a FIDO BBS in Brisbane. Does anybody else take this Echo??

The bottom line, however, is that we do need to maintain some communication with usergroups, firstly in USA, Europe and others, as well as have arrangements for updated and new software.

I would welcome any suggestions that would improve our present systems or reduce the present costs, and in particular if such suggestions could be presented to the OS9 Usergroups of the world, at the discussions in Chicago. Cheers, Gordon.

oooooooooooo0000000000oooooooooooo

Information about Floptical Drives
from the OS9 echo on FidoNet

From : Tom Birt
To : Timothy Mohr
Subject: Re: Floptical Drives

TM: Does anyone have any information on the new 'Floptical Drives'? I heard about it, and I would like to know more...

Timothy, since I heard that the Floptical drives were going to be available for OS-9, I have been doing some research;

In late January Mark Griffith announced on Delphi that a Floptical drive, with driver and descriptors, would be available for the MM/1 for \$350., including the hardware.

The only drive makers which I've seen advertising the Floptical (TM) drives are Iomega and PLI (Peripheral Land). The media is pre-formatted by 3M, Sony, and a few others (approx. \$20/disk) in packages of 5. They have the same dimensions as standard 3.5" floppies, but they weigh a few millio-unces less (I'm told). The media looks like translucent amber colored mylar, but is imbedded

with magnetic particles. The drives can read and write to 720K and 1.44M floppies (2.88M ?). Currently they can store 21MB of data. The FAT (File Allocation Table) is permanently etched, by laser into a VHD (Very High Density) format at the factory, so the FAT can't be accidentally destroyed. The access time is about the same as a slow hard drive - 65ms.

As far as I know, all the drives require a SCSI interface. The lowest retail price I heard of was \$300. The drive dimensions are the same as new floppy drives - 1/3 height.

These drives could take the place of many tape backups done now. There are rumors of 42M capacity in the near future, using the same drives, but with a 'stacker' interface. We now have a place to keep all those behemoth GIFs and archives. How nice it will be to mail one Floptical to a friend or associate, instead of a whole stack of floppies! And for only one first class stamp - 29 cents.

* Origin: The Byte Box, San Diego, CA 619-277-4618 (1:202/624)

oooooooooooo0000000000oooooooooooo

A C Tutorial
Chapter 13 - Character and Bit Manipulation
UPPER AND LOWER CASE

Load and display the program UPLOW.C for an example of a program that does lots of character manipulation. More specifically, it changes the case of alphabetic characters around. It illustrates the use of four functions that have to do with case. It should be no problem for you to study this program on your own and understand how it works. The four functions on display in this program are all within the user written function, "mix_up_the_line". Compile and run the program with the file of your choice. The four functions are;

```
isupper(); Is the character upper case?  
islower(); Is the character lower case?  
toupper(); Make the character upper case.  
tolower(); Make the character lower case.
```

CLASSIFICATION OF CHARACTERS

Load and display the next program, CHARCLAS.C for an example of character counting. We have repeatedly used the backslash n character representing a new line. There are several others

that are commonly used, so they are defined in the following table;

\n	Newline
\t	Tab
\b	Backspace
\"	Double quote
\\	Backslash
\0	NULL (zero)

By preceding each of the above characters with the backslash character, the character can be included in a line of text for display, or printing. In the same way that it is perfectly all right to use the letter "n" in a line of text as a part of someone's name, and as an end-of-line, the other characters can be used as parts of text or for their particular functions.

The program on your screen uses the functions that can determine the class of a character, and counts the characters in each class. The number of each class is displayed along with the line itself.

The three functions are as follows;

isalpha(); Is the character alphabetic?
 isdigit(); Is the character a numeral?
 isspace(); Is the character any of, \n, \t, or
 blank?

This program should be simple for you to find your way through so no explanation will be given. It was necessary to give an example with these functions used. Compile and run this program with any file you choose.

THE LOGICAL FUNCTIONS

Load and display the program BITOPS.C. The functions in this group of functions are used to do bitwise operations, meaning that the operations are performed on the bits as though they were individual bits. No carry from bit to bit is performed as would be done with a binary addition. Even though the operations are performed on a single bit basis, an entire byte or integer variable can be operated on in one instruction. The operators and the operations they perform are given in the following table;

& Logical AND, if both bits are 1, the result is 1.
 | Logical OR, if either bit is one, the result is 1.

^ Logical XOR, (exclusive OR), if one and only one bit is 1, the result is 1.

~ Logical invert, if the bit is 1, the result is 0, and if the bit is 0, the result is 1.

The example program uses several fields that are combined in each of the ways given above. The data is in hexadecimal format. It will be assumed that you already know hexadecimal format if you need to use these operations. If you don't, you will need to study it on your own. Teaching the hexadecimal format of numbers is beyond the scope of this tutorial.

Run the program and observe the output.

THE SHIFT INSTRUCTIONS

The last two operations to be covered in this chapter are the left shift and the right shift instructions. Load the example program SHIFTER.C for an example using these two instructions. The two operations use the following operators;

<< n Left shift n places.
 >> n Right shift n places.

Once again the operations are carried out and displayed using the hexadecimal format. The program should be simple for you to understand on your own, there is no tricky code.

oooooooooooooooooooooooooooooooo

FOR SALE FOR SALE FOR SALE

One complete set of manuals for access to CompuServe. This set of manuals would be invaluable to anyone who uses the US based Compuserve Information System, which has a multitude of OS9 and RSDOS files on it as well as informative discussion areas.

Price \$30.00 (o.n.o)

Contact:

Rob Mackay
 7 Harburg Drive
 BEENLEIGH. Qld. 4207

Phone: +61 7 8073802 (int'l)
 07 8073802 (Aust)

oooooooooooooooooooooooooooooooo

MPFM - Motion Picture File Manager
 A new product from Microware

[This review was taken from a publication from Microware Inc, sent to me by the Australian Microware distributor, Microprocessor Consultants Pty Ltd.]

Features

* Real-time audio and video playback of MPEG encoded files

- * Compatible with OS-9 and OS-9000 systems
- * ISO 11172 compliant
- * Video:
 - Full-screen, full motion video
 - Play forward, pause, frame step, loop, scan forward/reverse
 - Windowing, sizing, positioning
- * Audio:
 - Volume control, muting
 - Stereo mixing
 - Looping
- * Accepts input from optical, magnetic or RAM disks and networking
- * Device driver support for Motorola and C Cube MPEG decoders

Product Overview

The Motion Picture File Manager (MPFM) is a drop-in I/O extension for OS-9 and OS-9000 Operating Systems that supports real-time playback of MPEG encoded audio and video files. Users can now play back full-screen, full-motion video complete with synchronized digital audio.

MPFM supports world standard audio and video formats and provides options for playback control. Video can be configured to play back in a variety of display environments, including windowing, with full control of playback rates to scan, freeze, slow motion and frame step. Audio control provides volume control, stereo panning and muting.

MPFM Architecture

MPFM is a dual ported file manager written in C to accommodate OS-9 systems on Motorola 68XXX platforms and OS-9000 systems operating with 386/486 microprocessors. Like all Microware file managers, MPFM is 100 percent ROMable and can be easily added to any existing OS-9 or OS-9000 system.

Working example MPEG audio and video drivers are included to support playback on standard sound equipment and color monitors.

MPEG File Stream

The MPEG file stream consists of encoded video and audio packets multiplexed into multiple streams. MPFM supports up to 16 MPEG video and 32 MPEG audio streams. Either the MPFM device drivers or the MPEG hardware can demultiplex the streams. Channel switching during playback is supported at the driver level for seamless audio and video channel switching.

MPFM supports MPEG decoding data rates at 1.2M/sec. to 6M/sec. MPFM supports 24.25 and 30 frames per second with NTSC and PAL resolutions.

Audio/Video Synchronization

Sequenced audio and video may be played either independently or in a synchronized manner. Several requests can also be queued to sequentially play upon completion of the previous sequence.

MPFM Installation Pak

A MPFM Installation Pak provides all the modules needed to install and use MPFM on a single OS-9 or OS-9000 system. This package includes example driver source code for easy installation for popular MPEG decoders.

- * Motion Picture File Manager (MPFM) object code
- * MPFM video driver source code
 - C Cube CL450 video decoder
 - Motorola MCD250 video decoder
- * MPFM audio driver source code
 - Motorola MCD260 DSP
 - Analog Devices ADSP2105 DSP
- * Installation and User's manuals
- * 90-Day "Hotline" Support Registration Card

MPFM Board Support Pak

The MPFM Board Support Pak is a drop-in, binary code package for OS-9000 PC/AT systems supporting the C Cube Kermit MPEG decoder. This package combined with Microware's OS-9000 Development Pak instantly turns any 386/486 PC/AT into a real-time decoder allowing viewing and manipulation of MPEG encoded movies or files.

- * Motion Picture File Manager (MPFM)
- * MPFM video driver object code for C Cube CL450 video decoder
- * MPFM audio driver object code for Analog Devices ADSP2105 DSP
- * Installation and User's Manuals
- * 90-Day "Hotline" Support Registration Card

Hardware and Software Requirements

The MPFM Installation Pak can be installed on OS-9 V2.4 or OS-9000 V1.3 (or greater) systems.

The MPFM Board Support Pak for OS-9000 can be added to any 386/486 PC/AT supporting OS-9000 Development Pak V1.3 (or greater).

Ordering Information

Current single-copy pricing for MPFM Installation and Board Support Paks is found in the OS-9 and OS-9000 Software Products Price List.

To order your MPFM package or to obtain more information, contact your authorized Microware representative or call Microware at (515) 224-1929.

ooooooooooooOOOOOOOOOOoooooooooooo

Interleave tester
a PD programme from our library

I have chosen a programme from our PD collection to publish in this issue of the Newsletter. It is called 'Ileave', and is written in Basic09.

One of the options that an OS9 user has when he formats a disk is to change the disk's interleave. The interleave of standard OS9 disks is 3. This means that the sectors are physically spaced on the disk in this order:

1,7,13,2,8,14,3,9,15,4,10,16,5,11,17,6,12,18

This is done to allow the disk read software (CC3disk) more time to manipulate the data. For example it has to move it from the input buffer in system RAM to the actual programme RAM area. Bob van der Poel, the programme's author has been able to successfully change his interleave by examining data from this programme. Here are his comments:

This archive contains the following:

Readme — this file
Ileave — basic09 program
ILdata — results of run on my system

The whole point of this is to see what disk interleave factor (used when formatting a disk) is the most efficient. The BASIC09 program formats the disk with the various factors and then runs a series of read/write tests and times them.

NOTE: The program contains a number of SHELL "dsetime" statements. This resets my real time clock and ensures that the timing is accurate. If you have a no-halt controller these statements can be removed, if yours is a halt controller then you should either include a similar command or, if you delete it, be satisfied with inaccurate timings.

[Note that I have already REM'd them out and commented them. ED]

It seems from the results on my system that an interleave of 2 would be slightly more efficient

than the 3 the drive descriptors from Tandy contain.

Bob van der Poel [76510,2203]

April 1988.

Now here is the Basic09 code:

PROCEDURE ileave

REM Test various interleave factors on floppies and time

REM the results of various operations.

DIM path:BYTE
DIM cmd\$:STRING[60]
DIM t:INTEGER

```
SHELL "mode -pause"
CREATE #path,"/d0/ILdata":WRITE
FOR t=1 TO 17 \(* use all different factors
cmd$="format /dl r ""testdisk"" '10' 1
:"+STR$(t)+":
PRINT cmd$
SHELL cmd$
PRINT USING "'I-leave:',i3<",t;
PRINT #path USING "'I-Leave:',i3<",t;
FOR test=1 TO 3
READ cmd$
REM SHELL "dsetime" \(* removed this call to an
unknown module *)
time1$=DATE$
RUN cmd$
REM SHELL "dsetime" \(* removed this one too *)
time2$=DATE$
RUN time(time1$,time2$,elapsed)
PRINT USING "x3,s5<,':',i5<",cmd$,elapsed;
PRINT #path USING "x3,s5<,':',i5<",cmd$,elapsed;
NEXT test
PRINT
PRINT #path
NEXT t
CLOSE #path
```

DATA "test1","test2","test3"
PROCEDURE test1

REM write and read 3000 bytes to disk

```
DIM t:INTEGER
DIM char:BYTE
DIM path:BYTE
CREATE #path,"/dl/test1":UPDATE
FOR t=1 TO 3000
PUT #path,char
NEXT t
SEEK #path,0
FOR t=1 TO 3000
GET #path,char
NEXT t
CLOSE #path
PROCEDURE test2
REM read/write a 20000 byte array
```

```
DIM char(20000):BYTE
DIM path:BYTE
CREATE #path,"/dl/test2":UPDATE
PUT #path,char
SEEK #path,0
GET #path,char
CLOSE #path
PROCEDURE test3
REM load/save binary file to disk
REM this uses runb which is in memory and
REM is appx 12k long...

SHELL "save /dl/runb runb"
SHELL "load /dl/runb"
PROCEDURE time
REM Find elapsed time in seconds between 2 date$
```

PARAM time1\$,time2\$:STRING

PARAM elapsed:REAL

DIM Elap1,Elap2:REAL

```
Elap1=VAL(MID
$(time1$,10,2))*3600+VAL(MID$(time1$,13,2))*60+VAL(
MID$(time1$,16,2))
Elap2=VAL(MID
$(time2$,10,2))*3600+VAL(MID$(time2$,13,2))*60+VAL(
MID$(time2$,16,2))
elapsed=Elap2-Elap1
```

Here is a table of the author's results:

I-Leave: 0	test1: 26	test2: 37	test3: 26
I-Leave: 1	test1: 26	test2: 37	test3: 26
I-Leave: 2	test1: 25	test2: 8	test3: 9
I-Leave: 3	test1: 24	test2: 11	test3: 10
I-Leave: 4	test1: 25	test2: 10	test3: 10
I-Leave: 5	test1: 25	test2: 12	test3: 11
I-Leave: 6	test1: 24	test2: 16	test3: 13
I-Leave: 7	test1: 25	test2: 15	test3: 14
I-Leave: 8	test1: 26	test2: 18	test3: 14
I-Leave: 9	test1: 25	test2: 21	test3: 17
I-Leave: 10	test1: 26	test2: 21	test3: 17
I-Leave: 11	test1: 26	test2: 23	test3: 18
I-Leave: 12	test1: 26	test2: 25	test3: 19
I-Leave: 13	test1: 27	test2: 26	test3: 20
I-Leave: 14	test1: 26	test2: 28	test3: 22
I-Leave: 15	test1: 25	test2: 30	test3: 22
I-Leave: 16	test1: 26	test2: 32	test3: 24
I-Leave: 17	test1: 25	test2: 34	test3: 24

oooooooooooo0000000000oooooooooooo

Script1 - Shell+ v2.0 Scripts
by Steve Clark

The new version of Shell+ for OS-9 Level II on the Color Computer has introduced some powerful new capabilities for the Level II user. With the numerous extensions made by Ron Lammardo, along with Kevin Darling and Kent Myers, it is indeed possible for the user to "go wild" with shell scripts. I have only had the new shell+ a few days, but have converted some of my favorite shell programs to take advantage of some of the new features. I offer these as suggestions only, you will need to modify them to your liking. I did discover a few facts about the shell+ which I will mention as I go along.

This file contains a few shell+ scripts intended to demonstrate some of the things you can do with shell+. In some cases, I am sure there are alternate ways to do what these scripts do. Most of

these use other programs available on-line on various information services. I will try to mention these when they are encountered.

Ramdisk - Setup a Ramdisk

This shell+ script sets up the ramdisk, giving you a choice of what size. The rammer and /r0 ramdisk provided by Kevin Darling is used, and you must have these in your bootlist. You could modify the script for other ramdisk setups. It also uses Kevin's dnode utility, to set the size of the ramdisk (cyl=##). It uses the prompt and var.0 input, along with if/else/endif from the shell+. If you press ENTER, no ramdisk is created. Remember that once you create the ramdisk, you must reboot to get rid of it or change its size.

```
*ramdisk
prompt Ramdisk 1=40 track, 2=35 track, 3=20 track:
var.0
if %0=1
    echo 40 Track
    dmode /r0 cyl=28
else
if %0=2
    echo 35 Track
else
if %0=3
    echo 20 Track
    dmode /r0 cyl=14
else
    echo ....No Ramdisk Created....
    goto +end
endif
clrif
iniz /r0
format /r0 r "Ramdisk">>>/nil
echo /r0 created
free /r0
*end
```

Pctl - printer control

This sends display codes to the DMP-120 printer, setting various modes like condensed print, wide print, etc. It puts up a simple one-line menu for you to choose from, then sends the appropriate control codes to >/p. Change it for your printer and you won't have to look up the codes again. This one traps the error and prints a message when an error occurs (usually the printer has not been turned on).

```
*pctl
onerr goto errtrap
prompt Choose (n)ormal, (c)ondensed, (w)ide, (e)xtra
wide:
var.0
if %0=n
    display lb 0f lb 13>/p
else
if %0=c
    display lb 0f lb 14>/p
else
if %0=w
    display lb 14 lb 0e>/p
else
if %0=e
    display lb 13 lb 0e>/p
else
    echo Pctl terminated.
endif
clrif
goto +finis
*errtrap
```

```
echo An error %* occurred, check printer status.
*finis
```

Wctl - window control

Similar to ramdisk, this does various types of window setup. Some of the options work on the current window, some create new windows. I use other utilities like w80, gw80, gw40, etc. for quick window changes, but keep wctl for seldom-used window commands that I don't want to keep track of (like how to do an over & under 80 column setup) on paper. Again, modify it for your uses. This script simulates a multi-way if/then by actually nesting the if/then's but only using one endif and doing a final clrif to clear them out. This uses Fred Sawtelle's wmode utility. Also, I keep echo, display, and prompt in memory at all times; this speeds up execution.

```
*wctl
echo Pick one of the following:
echo 1 Graphics 640x192 (black on white 4 color)
echo 2 Graphics 320x192 (16 color)
echo 3 Text Window
echo 4 Over and Under W4 W5
echo 5 Graphics 640x192 (white on black 4 color)
prompt Choose:
var.0
if %0=1
    display lb 24 lb 20 7 0 0 50 18 2 0 0 lb 3a c8
01
    display lb 21 </1 >/1
else
if %0=2
    display lb 24 lb 20 8 0 0 28 18 0 1 0 lb 3a c8
02
    display lb 21 </1 >/1
else
if %0=3
    display lb 24
    display lb 20 2 0 0 50 18 0 2 3 lb 31 2 0 >/1
else
if %0=4
    wmode /w4 col=50 row=d wnd=4 val=1 sty=2 cpx=0
    cpy=0 fgc=0 bgc=2 bdc=3
    wmode /w5 col=50 row=a wnd=5 val=1 sty=ff cpx=0
    cpy=e fgc=0 bgc=2 bdc=3
    iniz /w4
    shell i=/w4&
    echo Go to w4 and start a shell for w5
else
if %0=5
    display lb 24 lb 20 7 0 0 50 18 0 2 2 lb 3a c8
01
    display lb 21 </1 >/1
endif
```

clrif

Env - environment setup

This is one of my favorites. The reason I wrote env was to facilitate rapid switching of environments. By "environment" I mean a particular combination of execution directory (cx), working directory (cd), etc. I used Ron's datamod to make this into a "mem" script which I keep in memory at all times. It is 960 bytes in the datamod version, a small price to pay for the convenience.

With my hard disk, I use a lot of subdirectories, for instance the subdirectory /DD/USR/PROG is where I put all my program development stuff (compilers, source code, test files, etc.). Under this I have a directory branch for each type of programming: C, ASM, BASIC09, SHELL programming, and so on. I keep the C compiler, for example, in /DD/USR/PROG/C/CMD5, and keep C source in /DD/USR/PROG/C. Similarly, I keep the assembler in /DD/USR/PROG/ASM/CMD5. This allows me to compile or assemble to the unique execution directory without messing up my working version of a program until the new version is finished. The problem this creates is that a lot of long path names have to be typed. If I am in C and want to refer to a Basic09 program, I have to type something like

```
list /dd/usr/prog/basic/file.b09
```

to list the file. If I'm not sure of the file name, I first have to do a dir on that directory. NO MORE! With the shell+ I can instantly switch to it by typing "env" and answering "b". I can get back to where I was by doing an "ex".

This is consistent with the windowing environment, using env something like the CLEAR key. It makes me more productive by saving a lot of path name typing. Unless your directories are exactly like mine, you can't use this directly, but can modify it to your liking. I experimented with this a lot before finally settling on the method of creating a new shell for the new environment. That avoids the DEAD shell syndrome, and only ties up one extra block of memory, plus I can get immediately back to where I came from. I also set a path=/DD/CMD5 or some alternate path so my main CMD5 is available when I need it. [NOTE: You can execute a program along a path by typing "edit xxxx" (for edit), but the load command doesn't know the path, so "load edit" doesn't work, but "load /dd/cmds/edit" does work.]

Env uses a computed "goto" which is not an

explicit part of shell+ but can be easily created by using the var.0 input as part of a goto, i.e., "goto +lab%0" puts the contents of var.0 at the end of "lab" before starting up. If you have a "*lab%0" label in the script, that is where it will go if you reply with "c". You need a "*" at the point where you want to catch non-matching input, and a "goto +somewhere" at the end of each branch to take you out.

I set a special prompt for each environment so I will know when I am in an "environment" vs. being in an original window shell.

```
*env
onerr goto +lab
prompt Choose - Asm Basic C Database Letter Other
Shell Terminal Vef Quit:
var.0
goto +lab%0
*lab%0
  cd /dd/usr/prog/asm
  cx /dd/usr/prog/asm/cmds
  var.2=/dd/cmds
  var.1=ASM>
  goto +cont
*lab%0
  cd /dd/usr/prog/basic
  cx /dd/usr/prog/basic/cmds
  var.1=B09>
  var.2=/dd/cmds
  goto +cont
*lab%0
  cd /dd/usr/prog/c
  cx /dd/usr/prog/c/cmds
  var.1=C>
  var.2=/dd/cmds
  goto +cont
*lab%0
  cd /dd/usr/data/db
  cx /dd/usr/data/db/cmds
  var.1=DB>
  var.2=/dd/cmds
  goto +cont
*lab%0
  cd /dd/usr/spc
  cx /dd/cmds
  var.1=LETTER>
  goto +cont
*lab%0
  cd /dd/usr/prog/other
  cx /dd/cmds
  var.1=OTHER>
  goto +cont
*lab%0
  cd /dd/usr/prog/script
  cx /dd/cmds
```

